AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

UNIVERSITY OF SOUTH CAROLINA

CSCE 240: Advanced Programming Techniques
Lecture 29: Wrap-up and Conclude

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

28TH APRIL 2022

Carolinian Creed: "I will practice personal and academic integrity."

Credits: Some material reused with permission of Dr. Jeremy Lewis. Others used as cited with thanks.

Organization of Lectures 29

- Introductory section
- Main section
  - Remaining project presentations
  - Recap
    - Course goals
    - Highlights
      - Lectures
      - Homework assignments and peer-evaluation
      - Programming assignment
  - Future: Programming, Research, AI
- Concluding section
  - Ask me anything

Introduction Section

Last Class

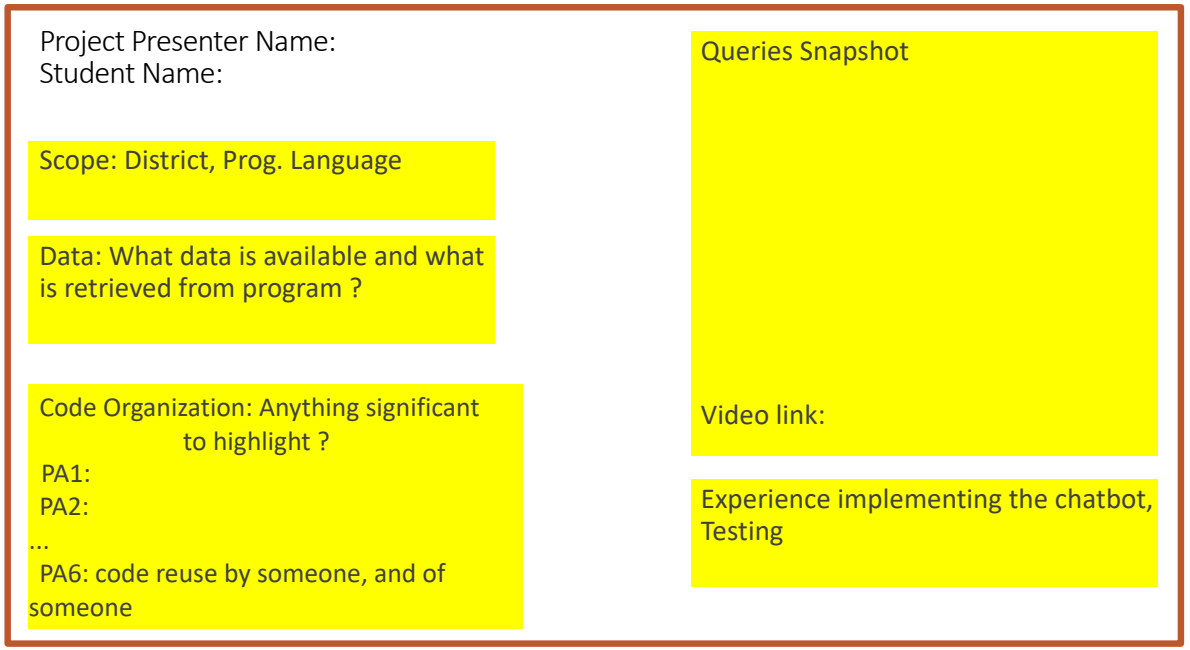
- Grades to be posted by Tuesday, May 3

Main Section

Remaining Project Presentations

Project Presentation Instructions

- Write name on sheet
- Use slides from your GitHub
- Take up to 3 mins to describe
- Answer questions up to 1 min
- Be present for others presentation



Course Wrap-Up

Learning Objectives

- Develop language-independent understanding of programming concepts by being exposed to multiple languages (C++, Java, Python)
- Independently design and implement programs in multiple language of choices (C++, Java or Python based on choice) in a Unix environment
- Demonstrate mastery of pointers, iterators, memory management including object creation and destruction, and parameter passing in C++
- Demonstrate mastery of object-oriented programming concepts including: inheritance, polymorphism, operator overloading, template functions and classes, and the use of STL containers.
- Develop object-oriented models using UML
- Able to work in programming teams with code review and walk throughs
- Solve practical problems that matter

Lectures: Topics Covered and In-Scope

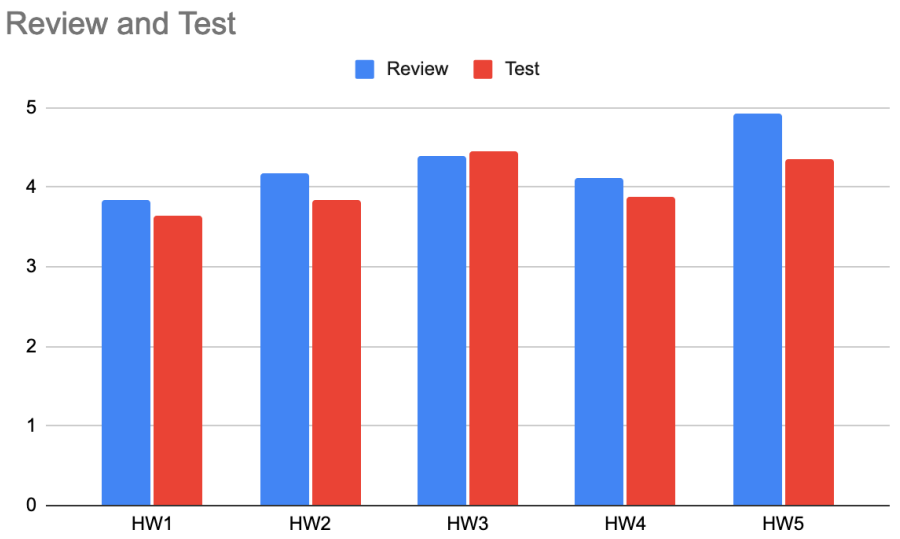
1	**Jan 11 (Tu)**	**Introduction**	
2	Jan 13 (Th)	Introduction – Pointers, Iteration	
3	Jan 18 (Tu)	Input/ Output	
4	Jan 20 (Th)	I/O, Exceptions	HW 1 due
5	Jan 25 (Tu)	Memory management, User defined types	Prog 1 - start
6	Jan 27 (Th)	Object Oriented (OO) intro	HW 2 due
7	Feb 1 (Tu)	OO concepts, UML Notations	
8	Feb 3 (Th)	Code org (C++)	Prog 1 - end
9	Feb 8 (Tu)	OO – inheritance	Prog 2 - start
10	Feb 10 (Th)	Regex, OO - polymorphism	HW 3 due
11	Feb 15 (Tu)	In class test	Quiz 1 – In class
12	**Feb 17 (Th)**	**Review: inheritance, Polymorphism**	
13	Feb 22 (Tu)	Exceptions	Prog 2 - end
14	Feb 24 (Th)	OO – Constructor, Destructor	Prog 3 - start
15	Mar 1 (Tu)	OO – operators, access control	HW 4 due
16	Mar 3 (Th)	C++ standard library	Prog 3 - end Semester - Midpoint
17	Mar 15 (Tu)	Testing strategies	Prog 4 - start
18	Mar 17 (Th)	Advanced: Pointers	HW 5 due
19	Mar 22 (Tu)	Advanced: Pointers, I/O	
20	Mar 24 (Th)	Advanced: Operator overloading	Prog 4 - end
21	Mar 29 (Tu)	Advanced: Memory Management	Prog 5 - start
22	Mar 31 (Th)	Advanced: Code efficiency	
23	Apr 5 (Tu)	Advanced: Templates	Prog 5 - end
24	Apr 7 (Th)	AI / ML and Programming	Prog 6 - assembling
27	**Apr 19 (Tu)**	**Project presentation**	
28	Apr 21 (Th)	Project presentation	Last day of class
29	**Apr 28 (Th)**	**Wrap-up and Conclusion**	**Examination**

Lecture Logistics

- Material on github
- Code in C++
  - Java and Python whenever feasible
- Homeworks (6) in C++; peer evaluated
- Prog. Assignments (6) in C++/Java/Python; TA and evaluated by instruction team
- Quizzes (2) – in-class and pseudo-code
- Other practices encouraged
  - Code reuse (bonus)
  - CASY (bonus), Codathon (bonus), McNair Fellowship

Homework Assignments and Peer-Evaluation

- HA1 to HW5, all in C++ (HA6 optional)
- At least 1 point improvement
  - Peer review (3.83 -> 4.92; 28.3% improvement)
  - Peer test (3.64 -> 4.35; 19.4% improvement)
- Min participation: 32
  - Max class strength: 50+
- Caveats
  - The students evaluated themselves although we gave the rubric on how to score.
  - The HWs were also on different topics.
  - The number of students participating in each quiz were not constant.



Still, the results are encouraging!

Many Interesting Insights

- Initiative to learn
  - Synonyms of terms, to detect intents better
  - Comparison at level of letters, to handle noisy text
  - Handling additional languages – Spanish
  - New UML diagramming tool – Mermaid - https://mermaid-js.github.io/mermaid/#/
  - …
- Choice of languages

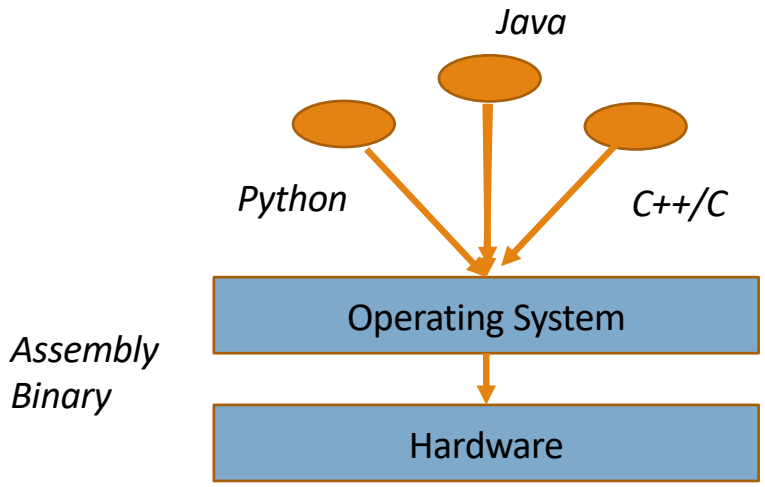
C++	12
Java	26
Python	6

Future: Programming, Research, AI

Programming Techniques

- Languages inevitably change over time
- Code practices remain
  - Adopt a language as mother tongue
    - Understand concepts in-depth
    - Experiment and settle on a coding style
  - Programming: variable initialization, understanding types, … usage of libraries
  - Memory: using just the right amount
  - Algorithms: focusing on efficiency
  - Documentation
  - Debugging methods
  - Testing, …

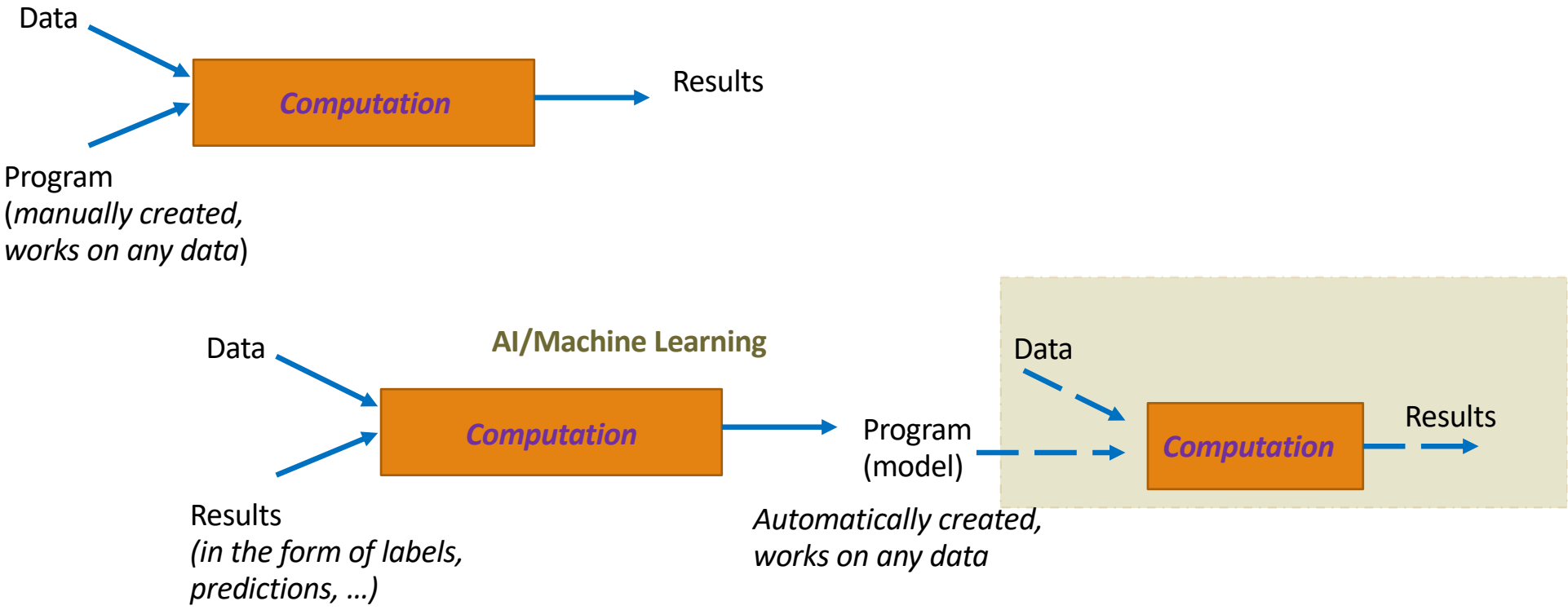
Programming languages are really for ***communicating among developers*** for building systems on OS/Hardware collaboratively



Research: Where is AI in IT?

- People have traditionally changed themselves to use IT
  - Examples: Typing, fixed menus
  - Focus on repeatability, user control
  - Disadvantage: usage barrier, entry barrier
- With AI: IT changing to enable people to use them naturally
  - Example: Natural language based interaction … chatbot
  - Focus on dynamicity, data-driven behavior
  - Disadvantage: hard to debug, audit and establish accountability

Traditional Programming v/s Machine Learning



Programming Trends

- Expect more languages to improve developer productivity
  - But good developers understand the underlying operating environment and have sound programming technique
- Expect more automatic code-generation
  - Example: OpenAI's Co-pilot: https://copilot.github.com/
- Automatic software generation is a long-established area
- AI in programming, and programming for AI will grow

Ask Me Anything